LAB5

1. Байгуулагч функц хэзээ дуудагддаг вэ?

Обьект үүсэх үед байгуулагч функц дуудагддаг.

2. Устгагч функц хэзээ дуудагдах вэ?

1. Үйлчлэх хүрээ дуусах үед/гарах үед local object-н устгагч функц дуудагдана.
2. Динамик санах ой ашиглан үүсгэсэн обьект-н нөөцөлсөн ойг delete-р устгахад дуудагдана.
3. Програм ажиллаж дуусах үед
4. Устгагч функцыг дуудах үед

3. Хуулагч байгуулагч гэж юу вэ? Ач холбогдол нь юу вэ?

Хуулагч байгуулагч бол шинэ обьект нь түүний өмнөх ямар нэг обьектын сүүлийн утгыг хуулбарлаж авах боломж олгох функц юм. Ямар нэг обьектыг үүсгэхдээ гарааны утганд нь өөр нэг обьектын сүүлийн хадгалагдаж байгаа утгыг оноож өгье гэвэл хуулагч байгуулагчийг ашиглана. Өөрөө тодохройлж өгөхгүй бол комфайлер үүсгэнэ.

4. Хуулагч функц гэж юу вэ? Ач холбогдол нь юу вэ? Санах ойн цоорхойгоос хэрхэн сэргийлэх вэ?

Хуулагч функц бол обьект-д түүний өмнөх ямар нэг обьектын сүүлийн утгыг хуулбарлаж авах боломж олгох функц юм. Энэ функц нь void төрөл байх ба ямар нэг утга буцаахгүй, public хэсэгт тодорхойлж байж л өөр object-г тухайн object-рүү хуулах боломжыг олгоно.

Хэрэв object-н гишүүн өгөгдөл дунд динамик санах ой ашиглан үүсгэсэн гишүүн байвал хуулхаасаа өмнө заавал утгатай эсэхийг шалгаж, байвал устгаж түүнийхээ дараах хуулах үйлдлийг хийх хэрэнтэй.

5. Объектын хаяган хувьсагчийг хэрхэн зарлах вэ? new операторoop санах ой нөөцлөх, хаяган хувьсагчаар дамжуулж объектын гишүүн өгөгдөл, гишүүн функцэд яаж хандах вэ?

```
class_name *Object_pointer_name;
Object_pointer_name = new class_name; (in our case new Employee)
// Шууд хандалт
(*object_name).function_name();
object_name->showData();
```

6. Объектын хаяган хувьсагчийн хүснэгт үүсгэх жишээ код бичиж ажиллуул.

```
#include <iostream>
using namespace std;
class worker{
    int number;
    float hours;
    string name, rank;
    float directorSalary(float salary){
            return salary+=500000;
        }
    public:
        //constructors
        worker(){
            number=0;
            name="";
            rank="";
            hours=0;
        }
        worker(int outNumber, const string outName,const string outRank,float outHours)
        {
```

```
            number=outNumber;
            name=outName;
            rank=outRank;
            hours=outHours;
        }
        worker(worker& employee){
            number=employee.number;
            name=employee.name;
            rank=employee.rank;
            hours=employee.hours;
        }
        void setEmployee(int num,string oName,string oRank,float oHours){
            number=num;
            name=oName;
            rank=oRank;
            hours=oHours;
        }
        void getEmployee(){
            cout<<"\nname : "<<name;
            cout<<"\nrank : "<<rank;
            cout<<"\nhours : "<<hours;
            cout<<"\nnumber : "<<number;
        }
        float salary(){
            float salary;
            salary=12000.0*hours;
            if(rank=="zahiral")salary+=directorSalary(salary);
            return salary;
        }
        bool timeIncreaser(float addedTime){
            if(addedTime>0 && addedTime<24){
                hours+=addedTime;
                return 1;
        }else{
                return 0;
} }};
int main() {
// Хүснэгт үүсгэх;
    worker employees[3];
    for(int i=0;i<3;i++){
        int number;
        float hours;
        string name, rank;
        cin>>number>>name>>rank>>hours;
        employees[i].setEmployee(number,name,rank,hours);
    }
    for(int i=0;i<3-1;i++){
        for(int j=0;j<3;j++){
            if(employees[j].salary<employees[i]){
                worker temp(employees[j]);
                employees[i] = employees[j];
                employees[j] = temp;
}
} }
```

```
    for(int i=0;i<3;i++){
        employees[i].getEmployee();
}
return 0; }
```

Бодлого

```
#include <iostream>
#include <cstring>
using namespace std;
class worker
{
    int number;
    float hours;
    char *name, rank;
    float directorSalary(float salary)
    {
        return salary += 500000;
    }

public:
    // unique id check function
    bool uniqueId(worker *w, int number)
    {
        for (int i = 0; i < (sizeof(w) / sizeof(*w)); i++)
        {
            if (w[i].number == number)
            {
                return false;
            }
        }
        return true;
    }
    // constructor
    worker()
    {
        this->number = 0;
        this->hours = 0;
        this->name = "";
        this->rank = '';
    }
    worker(int number, float hours, char *n, char *_rank)
    {
        // check if id is unique
        if (uniqueId(this, number))
        {
            cout << "id must be unique";
            return;
        }
```

```
        else
        {
            cout << "ID is not unique" << endl;
        }
        name = new char[strlen(n) + 1];
        strcpy(name, n);
        rank = new char[strlen(_rank) + 1];
        strcpy(rank, _rank);
        this->number = number;
        this->hours = hours;
    }
    // destructors
    ~worker()
    {
        delete[] name;
        delete[] rank;
    }
    // sort by name
    void sortByName(worker *w, int n)
    {
        worker temp;
        for (int i = 0; i < n; i++)
        {
            for (int j = i + 1; j < n; j++)
            {
                if (strcmp(w[i].name, w[j].name) > 0)
                {
                    temp = w[i];
                    w[i] = w[j];
                    w[j] = temp;
                }
            }
        }
    }
};
int main()
{
    worker gombo(1, 1, "name", "asdasd");
    return 0;
}
```